图文 31 RocketMQ生产集群准备：进行OS内核参数和JVM参数

详情 评论

RocketMQ生产集群准备：

进行OS内核参数和JVM参数的调整

石杉老哥重磅力作：《互联网java工程师面试突击》（第3季）**【强烈推荐】**：

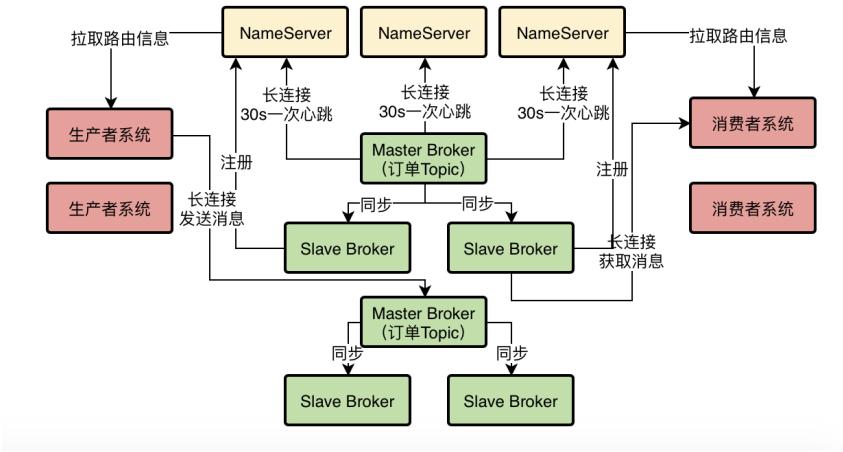
全程真题驱动，精研Java面试中**6大专题的高频考点**，从面试官的角度剖析面试

（点击下方蓝字试听）

《互联网Java工程师面试突击》（第3季）

1、压测前的准备工作

小猛在折腾了几天，用公司分配的几台机器部署好一个小规模的RocketMQ集群之后，终于开始思考如何进行压测的问题了。小猛盯着下面的架构图，陷入了一阵沉思，像RocketMQ这种中间件集群，应该如何进行压测，要准备哪些东西呢？



如果是一些经验不丰富的平时主要是做CRUD类的增删改查工作的Java工程师，可能此时就会直接尝试运行几个生产者和消费者的程序，然后多开一些线程写数据到RocketMQ，同时从RocketMQ消费数据

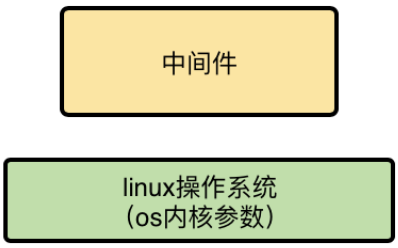
接着从RocketMQ的管理工作台中看一下TPS，每秒可以处理多少条消息，此时就算是完成压测了。

但是小猛并没有这样轻举妄动，他还是选择去咨询了一下明哥，提出了这个疑问，压测一个中间件真的就如此简单吗？

明哥听到小猛这个疑问后，就给小猛解释起了这种中间件集群的压测准备工作了。对于生产环境的中间件集群，不能直接用各种默认参数启动，因为那样可能有很多问题，或者没法把中间件的性能发挥出来。

对于一个中间件而言，第一步，你需要对他部署的机器的OS内核参数进行一定的调整（也就是linux操作系统的一些内核参数）

因为OS内核参数很多默认值未必适合生产环境的系统运行，有些参数的值需要调整大一些，才能让中间件发挥出来性能，我们看下面的的图。

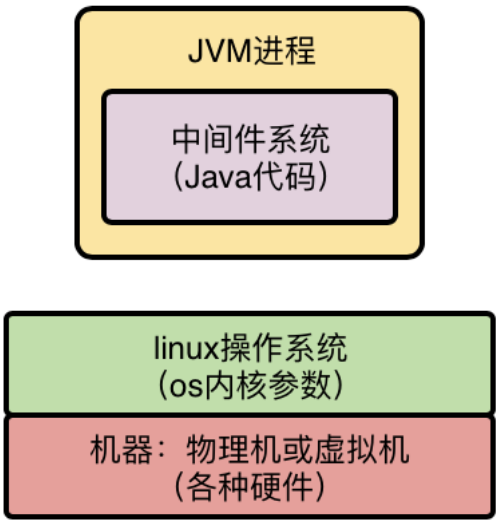


接着下一步需要思考的一个问题，就是一般中间件，比如RocketMQ、MyCat、Elasticsearch、Kafka之类的东西，很多都是Java开发的，或者是基于JVM的Scala开发的（比如Kafka）

所以你可以认为在一台机器上部署和启动一个中间件系统，说白了就是启动一个JVM进程，由这个JVM进程来运行中间件系统内的所有代码，然后实现中间件系统的各种功能。

我们看下面的图，这里清晰的展示出了一台机器上部署了一个中间件系统，就是启动了一个JVM进程的概念。

这里如果有朋友对JVM这块知识不太了解，给大家推荐一下狸猫技术窝上的《从0开始带你成为JVM实战高手》这个专栏，是我的好朋友救火队队长写的，已经更新完毕。



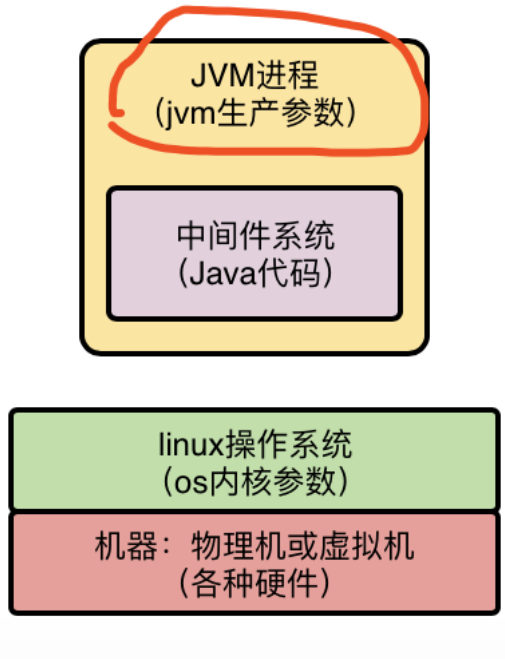
所以其实对于一个生产环境的中间件系统而言，在部署和启动之前，需要关注的第二个东西就是JVM的各种参数

比如内存区域的大小分配，垃圾回收器以及对应的行为参数，GC日志存放地址，OOM自动导出内存快照的配置，等等。

所以你就需要对JVM进行合理的优化配置，比如最简单的一点，明明你部署了一个几十GB内存的高配置物理机，结果你就给中间件系统的JVM分配了1GB的内存，你觉得这是不是在开玩笑?

相当于你机器配置很高，结果你的中间件系统就用了里面的几十分之一的内存，根本没用上那么多的资源!

我们看下面的图画圈的地方，这里就引申出了要配置JVM参数的概念，这是很多人都会忽略的一点。



最后第三件事情，就是中间件系统自己本身的一些核心参数的设置，比如你的中间件系统会开启很多线程处理请求和工作负载，然后还会进行大量的网络通信，同时会进行大量的磁盘IO类的操作。

这个时候你就需要依据你的机器配置，合理的对中间件系统的核心参数进行调整

比如你的机器配置很高，是24核CPU，结果你的中间件系统默认就开启了4个工作线程去处理请求，这不是在开玩笑么！相当于24核CPU里很多都是空闲状态，是没有任何事情可以干的。

要是不进行合理的参数设置，几乎可以认为就是在浪费高配置的机器资源!

所以以上三点，就是对任何一个中间件系统，在进行压力测试以及生产环境部署之前，都必须要进行调整的!

当然如果是普通的那种Java Web业务系统，通常而言上线之前主要关注的就是JVM的参数而已，对os内核参数以及业务系统自身参数大多数情况下都没有太多的要求

但是中间件系统而言，往往必须要对**os内核参数、jvm参数以及自身核心参数**都做出相对应的合理的调整，再进行压测和上线。

2、对RocketMQ集群进行OS内核参数的调整

接着明哥就开始为小猛讲解RocketMQ集群部署的机器需要调整的一些os内核参数的含义，并且给小猛说了一些他建议的调整值。

（1）vm.overcommit_memory

“vm.overcommit_memory”这个参数有三个值可以选择，0、1、2。

如果值是0的话，在你的中间件系统申请内存的时候，os内核会检查可用内存是否足够，如果足够的话就分配内存给你，如果感觉剩余内存不是太够了，干脆就拒绝你的申请，导致你申请内存失败，进而导致中间件系统异常出错。

因此一般需要将这个参数的值调整为1，意思是把所有可用的物理内存都允许分配给你，只要有内存就给你来用，这样可以避免申请内存失败的问题。

比如我们曾经线上环境部署的Redis就因为这个参数是0，导致在save数据快照到磁盘文件的时候，需要申请大内存的时候被拒绝了，进而导致了异常报错。

可以用如下命令修改：echo 'vm.overcommit_memory=1' >> /etc/sysctl.conf。

（2）vm.max_map_count

这个参数的值会影响中间件系统可以开启的线程的数量，同样也是非常重要的

如果这个参数过小，有的时候可能会导致有些中间件无法开启足够的线程，进而导致报错，甚至中间件系统挂掉。

他的默认值是65536，但是这个值有时候是不够的，比如我们大数据团队的生产环境部署的Kafka集群曾经有一次就报出过这个异常，说无法开启足够多的线程，直接导致Kafka宕机了。

因此建议可以把这个参数调大10倍，比如655360这样的值，保证中间件可以开启足够多的线程。

可以用如下命令修改：echo 'vm.max_map_count=655360' >> /etc/sysctl.conf。

（3）vm.swappiness

这个参数是用来控制进程的swap行为的，这个简单来说就是os会把一部分磁盘空间作为swap区域，然后如果有的进程现在可能不是太活跃，就会被操作系统把进程调整为睡眠状态，把进程中的数据放入磁盘上的swap区域，然后让这个进程把原来占用的内存空间腾出来，交给其他活跃运行的进程来使用。

如果这个参数的值设置为0，意思就是尽量别把任何一个进程放到磁盘swap区域去，尽量大家都用物理内存。

如果这个参数的值是100，那么意思就是尽量把一些进程给放到磁盘swap区域去，内存腾出来给活跃的进程使用。

默认这个参数的值是60，有点偏高了，可能会导致我们的中间件运行不活跃的时候被迫腾出内存空间然后放磁盘swap区域去。

因此通常在生产环境建议把这个参数调整小一些，比如设置为10，尽量用物理内存，别放磁盘swap区域去。

可以用如下命令修改：echo 'vm.swappiness=10' >> /etc/sysctl.conf。

（4）ulimit

这个是用来控制linux上的最大文件链接数的，默认值可能是1024，一般肯定是不够的，因为你在大量频繁的读写磁盘文件的时候，或者是进行网络通信的时候，都会跟这个参数有关系

对于一个中间件系统而言肯定是不能使用默认值的，如果你采用默认值，很可能在线上会出现如下错误：error: too many open files。

因此通常建议用如下命令修改这个值：echo 'ulimit -n 1000000' >> /etc/profile。

（5）一点小小的总结

其实大家综合思考一下这几个参数，会发现到最后要调整的东西，无非都是跟磁盘文件IO、网络通信、内存管理、线程数量有关系的，因为我们的中间件系统在运行的时候无非就是跟这些打交道。

中间件系统肯定要开启大量的线程**（跟vm.max_map_count有关）**
而且要进行大量的网络通信和磁盘IO**（跟ulimit有关）**
然后大量的使用内存**（跟vm.swappiness和vm.overcommit_memory有关）**

所以对OS内核参数的调整，往往也就是围绕跟中间件系统运行最相关的一些东西。

3、对JVM参数进行调整

接着明哥开始给小猛讲解起了RocketMQ的JVM参数如何调整，首先得先找到RocketMQ启动脚本中是如何设置JVM参数的，并且明白默认JVM参数的含义，同时再有针对性的做出适当的调整和修改。

我们回顾一下之前讲部署RocketMQ集群的时候，给大家介绍过几个启动脚本

在rocketmq/distribution/target/apache-rocketmq/bin目录下，就有对应的启动脚本，比如mqbroker是用来启动Broker的，mqnamesvr是用来启动NameServer的。

用mqbroker来举例，我们查看这个脚本里的内容，最后有如下一行：

```
sh ${ROCKETMQ_HOME}/bin/runbroker.sh org.apache.rocketmq.broker.BrokerStartup $@
```

这一行内容就是用runbroker.sh脚本来启动一个JVM进程，JVM进程刚开始执行的main类就是org.apache.rocketmq.broker.BrokerStartup

我们接着看runbroker.sh脚本，在里面可以看到如下内容：

```
JAVA_OPT="${JAVA_OPT} -server -Xms8g -Xmx8g -Xmn4g"
JAVA_OPT="${JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -
XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRefLRUPolicyMSPerMB=0"
JAVA_OPT="${JAVA_OPT} -verbose:gc -Xloggc:/dev/shm/mq_gc_%p.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -
XX:+PrintGCApplicationStoppedTime -XX:+PrintAdaptiveSizePolicy"
JAVA_OPT="${JAVA_OPT} -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m"
JAVA_OPT="${JAVA_OPT} -XX:-OmitStackTraceInFastThrow"
JAVA_OPT="${JAVA_OPT} -XX:+AlwaysPreTouch"
JAVA_OPT="${JAVA_OPT} -XX:MaxDirectMemorySize=15g"
JAVA_OPT="${JAVA_OPT} -XX:-UseLargePages -XX:-UseBiasedLocking"
JAVA_OPT="${JAVA_OPT} -Djava.ext.dirs=${JAVA_HOME}/jre/lib/ext:${BASE_DIR}/lib"
#JAVA_OPT="${JAVA_OPT} -Xdebug -Xrunjdwp:transport=dt_socket,address=9555,server=y,suspend=n"
JAVA_OPT="${JAVA_OPT} ${JAVA_OPT_EXT}"
JAVA_OPT="${JAVA_OPT} -cp ${CLASSPATH}"
```

在上面的内容中，其实就是在为启动Broker设置对应的JVM参数和其他一些参数，我们可以把其中JVM相关的参数抽取出来给大家解释一下：

```
“-server -Xms8g -Xmx8g -Xmn4g -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -
XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRefLRUPolicyMSPerMB=0 -verbose:gc -Xloggc:/dev/shm/mq_gc_%p.log -
XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCApplicationStoppedTime -XX:+PrintAdaptiveSizePolicy -
XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m -XX:-OmitStackTraceInFastThrow -
XX:+AlwaysPreTouch -XX:MaxDirectMemorySize=15g -XX:-UseLargePages -XX:-UseBiasedLocking”
```

我们来分门别类解释一下这些参数，当然解释这些参数是建立在大家要对JVM有一定的了解基础之上，因此还是建议大家先看一下狸猫技术窝的《从0开始带你成为JVM实战高手》这个专栏。

-server：这个参数就是说用服务器模式启动，这个没什么可说的，现在一般都是如此

-Xms8g -Xmx8g -Xmn4g：这个就是很关键的一块参数了，也是重点需要调整的，就是默认的堆大小是8g内存，新生代是4g内存，但是我们的高配物理机是48g内存的

所以这里完全可以给他们翻几倍，比如给堆内存20g，其中新生代给10g，甚至可以更多一些，当然要留一些内存给操作系统来用

-XX:+UseG1GC -XX:G1HeapRegionSize=16m：这几个参数也是至关重要的，这是选用了G1垃圾回收器来做分代回收，对新生代和老年代都是用G1来回收

这里把G1的region大小设置为了16m，这个因为机器内存比较多，所以region大小可以调大一些给到16m，不然用2m的region，会导致region数量过多的

-XX:G1ReservePercent=25：这个参数是说，在G1管理的老年代里预留25%的空闲内存，保证新生代对象晋升到老年代的时候有足够空间，避免老年代内存都满了，新生代有对象要进入老年代没有充足内存了

默认值是10%，略微偏少，这里RocketMQ给调大了一些

-XX:InitiatingHeapOccupancyPercent=30：这个参数是说，当堆内存的使用率达到30%之后就会自动启动G1的并发垃圾回收，开始尝试回收一些垃圾对象

默认值是45%，这里调低了一些，也就是提高了GC的频率，但是避免了垃圾对象过多，一次垃圾回收耗时过长的问题

-XX:SoftRefLRUPolicyMSPerMB=0：这个参数默认设置为0了，在JVM优化专栏中，救火队队长讲过这个参数引发的案例，其实建议这个参数不要设置为0，避免频繁回收一些软引用的Class对象，这里可以调整为比如1000

-verbose:gc -Xloggc:/dev/shm/mq_gc_%p.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCApplicationStoppedTime -XX:+PrintAdaptivePolicy -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m：这一堆参数都是控制GC日志打印输出的，确定了gc日志文件的地址，要打印哪些详细信息，然后控制每个gc日志文件的大小是30m，最多保留5个gc日志文件。

-XX:-OmitStackTraceInFastThrow：这个参数是说，有时候JVM会抛弃一些异常堆栈信息，因此这个参数设置之后，就是禁用这个特性，要把完整的异常堆栈信息打印出来

-XX:+AlwaysPreTouch：这个参数的意思是我们刚开始指定JVM用多少内存，不会真正分配给他，会在实际需要使用的时候再分配给他

所以使用这个参数之后，就是强制让JVM启动的时候直接分配我们指定的内存，不要等到使用内存的时候再分配

-XX:MaxDirectMemorySize=15g：这是说RocketMQ里大量用了NIO中的direct buffer，这里限定了direct buffer最多申请多少，如果你机器内存比较大，可以适当调大这个值，如果有朋友不了解direct buffer是什么，可以自己查阅一些资料。

-XX:-UseLargePages -XX:-UseBiasedLocking：这两个参数的意思是禁用大内存页和偏向锁，这两个参数对应的概念每个要说清楚都得一篇文章，所以这里大家直接知道人家禁用了两个特性即可。

最后我们做一点小的总结，RocketMQ默认的JVM参数是采用了**G1垃圾回收器，默认堆内存大小是8G**

这个其实完全可以根据大家的机器内存来调整，你可以增大一些也是没有问题的，然后就是一些G1的垃圾回收的行为参数做了调整，这个一般我们不用去动，然后就是对GC日志打印做了设置，这个一般也不用动。

其余的就是禁用一些特性，开启一些特性，这些都直接维持RocketMQ的默认值即可。

4、对RocketMQ核心参数进行调整

明哥讲完RocketMQ默认的JVM参数以及我们要做的一些微调之后，就继续讲RocketMQ自身的一些核心参数的调整了。

之前讲解集群部署的时候给大家提过，在下面的目录里有dledger的示例配置文件：rocketmq/distribution/target/apache-rocketmq/conf/dledger

在这里主要是有一个较为核心的参数：**sendMessageThreadPoolNums=16**

这个参数的意思就是RocketMQ内部用来发送消息的线程池的线程数量，默认是16

其实这个参数可以根据你的机器的CPU核数进行适当增加，比如机器CPU是24核的，可以增加这个线程数量到24或者30，都是可以的。

RocketMQ还有一些其他的核心参数，在后续专栏讲解的过程中，咱们再继续来分析如何优化和调整。

5、今日内容总结

我们最后来简单对今天讲解的内容作一下总结：

（1）中间件系统在压测或者上生产之前，需要对三大块参数进行调整：**OS内核参数、JVM参数以及中间件核心参数**

（2）OS内核参数主要调整的地方都是跟磁盘IO、网络通信、内存管理以及线程管理有关的，需要适当调节大小

（3）JVM参数需要我们去中间件系统的启动脚本中寻找他的默认JVM参数，然后根据机器的情况，对JVM的堆内存大小，新生代大小，Direct Buffer大小，等等，做出一些调整，发挥机器的资源

（4）中间件核心参数主要也是关注其中跟网络通信、磁盘IO、线程数量、内存 管理相关的，根据机器资源，适当可以增加网络通信线程，控制同步刷磁盘或者异步刷磁盘，线程数量有多少，内存中一些队列的大小

End

狸猫技术窝其他精品专栏推荐：

《从零开始带你成为JVM实战高手》

《21天Java 面试突击训练营》（分布式篇）（现更名为：**互联网Java工程师面试突击第2季**）

互联网Java工程师面试突击（第1季）

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情在评论区留言提问，我会逐一答疑

如何加群：购买了狸猫技术窝专栏的小伙伴都可以加入**狸猫技术交流群**

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群？》**（购买后可见）**